# CS-5540 - Gene Sequence Vectorization and Analysis

## Midterm Report

**Group 1**
Tom Steinman
Thomas Jones

| | |
|---|---|
| This document | https://docs.google.com/document/d/1rdYE9IYkq4M9juYEkHD5IvTRmP3WqDhMDj-g__Jfj5Q/edit?tab=t.0#heading=h.gkbj0e3grkgu |
| Dataset | https://huggingface.co/datasets/arcinstitute/opengenome2 |
| Source | https://colab.research.google.com/drive/1wyinSNAvUtpudUq_7yLNeBXfWk1flfCq#scrollTo=WXkfVU9nDu7c |

# Overview

The goal of the project is to design a scalable gene sequence search and analysis system using Big Data tools. We are analyzing a large genomic dataset to identify frequently occurring k-mers (short DNA subsequences) and codon (sequence of 3 nucleotides) usage patterns across diverse organisms. This allows us to both study biological sequence composition and demonstrate distributed MapReduce computation at scale. As a secondary goal, we hope to perform codon usage bias analysis, computing frequencies of codons and comparing their relative usage across species. This will let us identify evolutionary and translational patterns and complement our k-mer frequency analysis and search.

The key goals defined for the project are:

- Using the Open Genome 2 dataset ingest the selected organelle DNA sequences.
- Perform k-mer frequency analysis (k = 3+) and codon usage bias computation.
- Visualize results using GC% histograms, top-k k-mer charts, and codon usage heatmaps.
- Measure runtime and scalability (5 GB - 10 GB dataset) on a distributed Spark/Hadoop cluster.
- Prototype subsequence search using vectorized representations for approximate matching. (Codon frequency)

# Project Status

Google Colab was selected as the initial implementation environment. While the initial project plan did call for the use of the Hellbender cluster the setup uplift required conflicted with a desire for quick prototyping, something that Colab is special purpose built for.

The implementation consists of a 3 stage ingest pipeline:

1. Download the dataset from HuggingFace. The base file is a gzipped fasta formatted sequence.
2. Ingest the data into Parquet using a fasta library, Bio.SeqIO. For each record in the fasta file a new Parquet row is created specifying the sequence id, a description (if present), the length of the sequence, the actual DNA sequence, and finally a source indicating the origin source of the data (in this case all organelle).
3. Spark is initialized using the Parquet files. The data is ingested as whole string rows of the form {seq_id, description, length, sequence}. We chose not to split the sequences at this time because finding k-mers requires scanning across the entire string.

Once the data is ingested and Spark started we are able to query the data though our k-mer length is limited currently due to computing limitations. We are able to identify 6-mer counts across the entire loaded dataset. We are also able to construct codon (3-mer) frequencies.

We have explored three different search mechanisms:

1. Depth First Search - We perform a scan across the sequences starting with the first letter from the search string and proceeding across the full sequence. This is a brute force approach as it requires a full scan of every sequence.
2. TF-IDF Cosine Search - This is a natural language search algorithm utilizing the frequency of a given word across a corpus of documents as well as within a document. The cross-document frequency is meant to be a negative indicator in the text search domain. The search is fast though we would need to explore the results in more detail and with a bigger dataset.
3. Cosine Similarity Search - Following similar patterns as used in text encoding we also computed the sequences as directional vectors and calculated the cosine similarity to determine “closest” sequence to a given search string. The results are promising though need to continue to be explored.

The key with the last two algorithms is that they return “near” matches, not exact. It is possible we could use them as an initial pre-filter step to narrow down the total search space but that is a future direction.

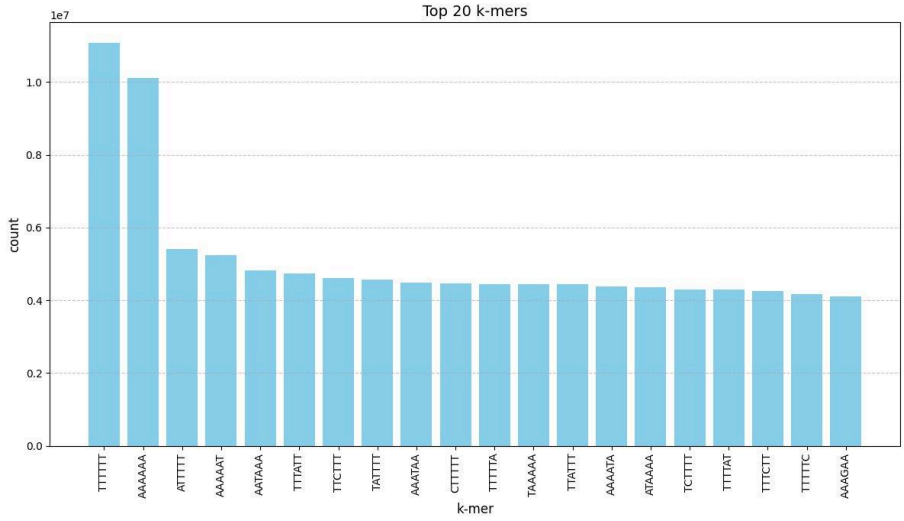
# Current Results

Loaded data from original files

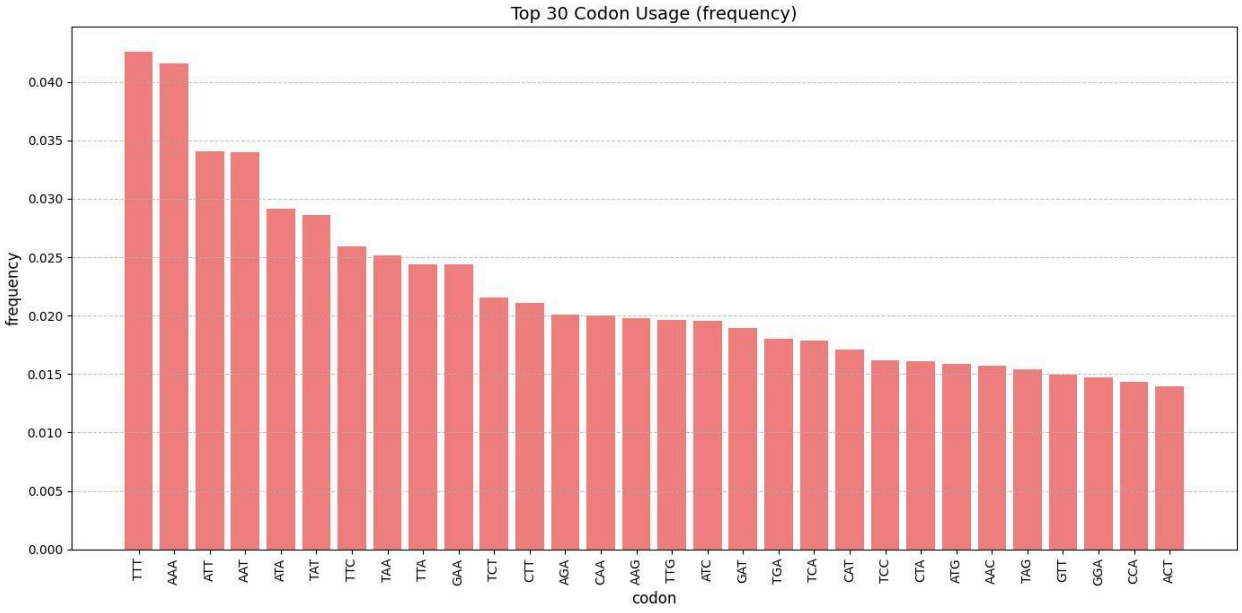
Rows in example shard: 1000

| | seq_id | description | length | sequence | source |
|---|---|---|---|---|---|
| **38** | NC_001322.1 | NC_001322.1 | 16019 | AATGAATTGCCTGATAAAAAGGGTTACCTTGATAGGGTAAATTATG... | organelle |
| **153** | NC_002609.1 | NC_002609.1 | 17019 | AATAAGGTGCCTGATAAAAGGGCTATCTTGATAGGATAGATAATGT... | organelle |
| **838** | NC_008097.1 | NC_008097.1 | 184933 | AAATAGGTAGTCTGTAATTTCAATAAATAAAAAAAAATAACAAAGAA... | organelle |

Organelle 6-mer top 20




Codon top 30



# Challenges and Next Steps

There are two primary challenges being faced. First, while Colab does provide a convenient environment for execution the underlying infrastructure can be opaque. The recent Spark update to 4.0 broke the installed version of Java on Colab. This was fixed by manually installing Java 17 at the start of the script.

The second challenge has been memory tuning. Within the current environment, Spark consistently runs into java OutOfMemory errors. This has been addressed temporarily by limiting the amount of data being processed. This is a temporary fix and the full solution will be addressed by final project completion. It may be necessary to continue the work within another execution environment.

The next steps for the project are; first, solidify scale. The current dataset is limited to simple organisms and for demonstration purposes they are not that exciting as the names have little or no real world meaning. However, the lengths of even simple insects are orders of magnitude larger than the current organelle dataset, for example, the fruit fly genome contains more than 150 million base pairs. The current sequences we are loading, by contrast, are in the hundreds of thousands.

Second, as related to scale as well, we will finalize our search mechanism. Ideally, at scale, we will be able to locate k-mers across a wide variety of sequences and species with only horizontal scale cost as the limitation (meaning we can’t afford a cluster of 1000 search servers). We will further explore ways to “index” into the data so that a first pass can eliminate as many sequences as possible before performing exhaustive scans.

Finally, we will polish the graphics and integrate those into the overall search application.

# Attribution of Work

Tom Steinman - Majority of code, presentations
Thomas Jones - Reports, debugging